بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

GU TECH

Data Structures

Lab 12

December 30, 2025

Contents

Lab Tasks

1. Implement a `Graph` class using both adjacency matrix and adjacency list representations, and implement the following:

   - Warshall's algorithm for accessibility.
   - Floyd's all pairs shortest path algorithm with **path tracking**.
     - Your code should contain a function that takes the two output matrices and any two nodes, and prints the shortest path between them if there is any.
   - Dijkstra's single source shortest path algorithm.
     - Your code should contain a function that takes the output of Dijkstra's algorithm and any two nodes, and prints the shortest path between them if there is any.
   - BFS based single source shortest path algorithm.
   - Prim's and Kruskal's algorithms for minimum spanning tree.

   Requirements:

   - Users should be able to select the representation (adjacency matrix or adjacency list) at the time of object creation.
   - Users should not be required to provide names of nodes. If no names are provided, the class should assign single letter names starting from 'A' until 'Z', at least for the first 26 nodes.

   Instructions:

   - There is no starter code.
   - Algorithm functions do not necesarily need to be members of a class.
   - Submit screenshots of the test cases along with your code files.

Testing:

- For Floyd Warshall algorithm, use the given graph. Your distance and predecessor matrices should be:

0	2	4	3
3	0	2	6
2	4	0	4
-2	0	2	0

-	A	B	A
B	-	B	A
D	A	-	C
D	A	B	-

  As such, the path from `C` to `B`, for example, should be: `C -> D -> A -> B`.

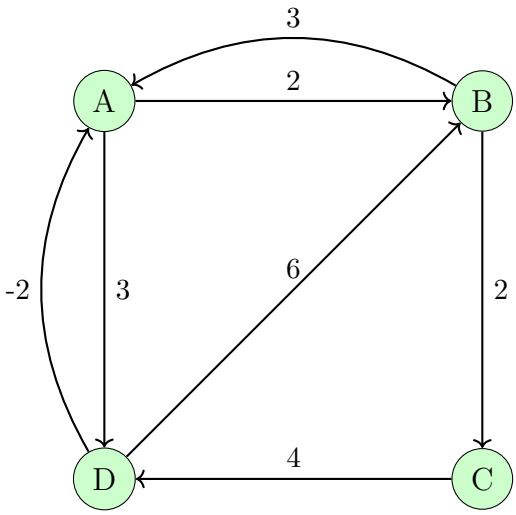

Figure 1: Directed graph for testing Floyd-Warshall algorithm

Graph source

- For Dijkstra's algorithm, use the given graph and return the following output matrix (assuming that the source node is `A`):

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

As such, the path from `A` to `C`, for example, should be: `A -> B -> D -> F -> C` for a cost of 12 units.

Note: Your output does not necessarily need to be a 3×1 matrix. You are free to format your output in any way you like, as long as it contains the required information.

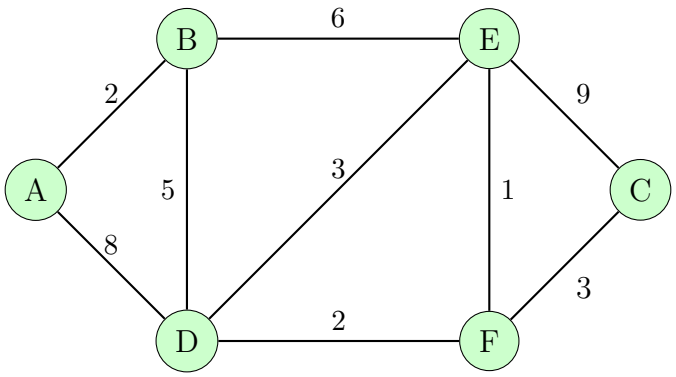

Figure 2: Undirected graph for testing Dijkstra's algorithm

Graph source

- For Prim's algorithm, use the given graph and return the following output matrix (assuming that the source node is `A`):

Edge	Weight Distance
A - B	2
A - C	3
A - D	3
C - E	1
C - F	6
F - G	9

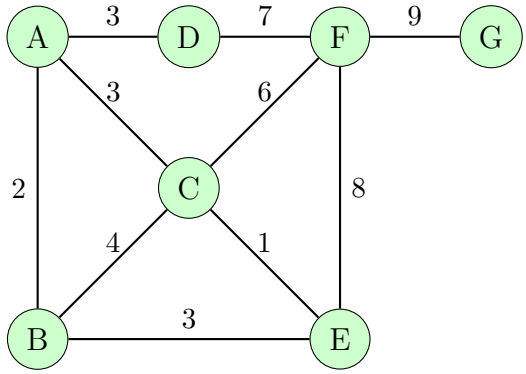

Figure 3: Undirected graph for testing Prim's algorithm

Graph source

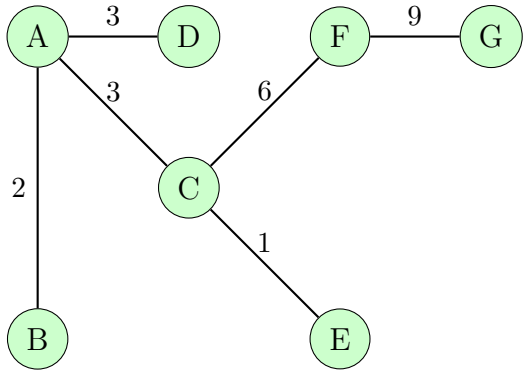

Figure 4: MST by Prim's algorithm for the given graph

- For Kruskal's algorithm, use the given graph and return the following output matrix (assuming that the source node is `A`):

Edge	Weight Distance
A - B	2
A - C	3
A - D	3
C - E	1
D - F	7
F - G	9

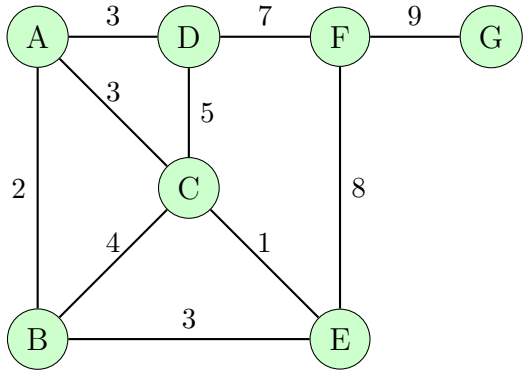

Figure 5: Undirected graph for testing Kruskal's algorithm

Graph source

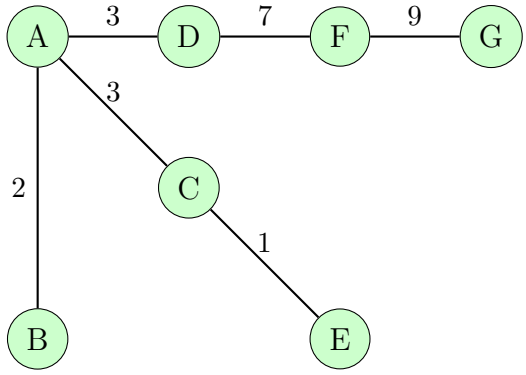

Figure 6: MST by Kruskal's algorithm for the given graph